Project Report

Modern Application Development - I

Author
Soham S. Kulkarni
22f3002597
22f3002597@ds.study.iitm.ac.in

Personal Introduction

I am a full time BS degree student, a tech enthusiast and an Indian classical music practitioner. My main interests include developing data-driven applications, developing AI/ML models and reading about new research in the field. Python is the basis for the majority of my work. This project was especially a lot of fun to work on, I learnt many new things and gained a lot of experience. There were many challenges that I had to overcome which was the most important part in my learning process.

Project Description

The project discussed below aims to provide an interface for an e-library (i.e. a Library Management System). This application can be used by both users and librarians. Users can request, read and buy books (PDF files) with some restrictions as specified by the guideline document (The user can request utmost 5 books at a time and also a user can access a book maximum for 15 days after which access to the book will be revoked). The user also has a cart functionality through which he can buy many books in a single click. On the other a librarian has more authority such as updating/adding books, sections, authors and accessing some statistics about the application. Both users and the librarian have separate login systems where a user can sign up too, RBAC is used to differentiate between user and librarian. This app is developed keeping in mind factors such as ease of access, minimalistic styling and minimal complexities (like adding buttons and links only when required, easy search etc.). For the storage purpose sqlite3 has been used but for storage of book content a different approach has been used wherein the pdf files are directly stored in the server and only the filename is stored in the database, this allows ease of displaying book content and downloading the book as pdf too. Various models are used in order to store different information about different components of the app. As part of the project a REST API's are developed for the Books, Sections, Authors and Graphs to get, update, add or delete book, section or author. The graph API has only a GET method allowed so graphs can be fetched using it.

Technologies/Modules/Libraries used

HTML: Used for creating templates
CSS: Used for styling web pages
Bootstrap: A simple yet powerful framework used for styling minimalistic web pages
Flask: An python library for developing backend of the web applications
Flask-RESTful: An extension for flask used to develop API's
Flask-SQLAlchemy: An extension for flask used to integrate SQLAlchemy with flask

SQLAlchemy: It is an open-source SQL toolkit and object-relational mapper (ORM) for the python
Jinja2: It is a web template engine for python
SQLite: It is a database engine

Controllers

Below controllers are used as per the requirements:

- @app.route("/", methods=['GET','POST'])
- @app.route("/lib/login", methods=['GET','POST'])
- @app.route("/user/signup", methods=['GET','POST'])
- @app.route("/<int:user_id>/home")
- @app.route("/<int:user_id>/profile", methods=['GET' , 'POST'])
- @app.route("/<int:user_id>/books")
- @app.route("/<int:user_id>/user/return_book/<int:book_id>")
- @app.route("/<int:user_id>/books/<int:book_id>")
- @app.route("/<int:user_id>/request/<int:book_id>")
- @app.route("/<int:user_id>/books_req")
- @app.route("/<int:user_id>/user/withdraw_req/<int:book_id>")
- @app.route("/lib/<int:user_id>/profile", methods=['GET' , 'POST'])
- @app.route("/<int:user_id>/all_books")
- @app.route("/<int:book_id>/remove_book")
- @app.route("/user/<int:user_id>/<int:book_id>/update_book", methods=['GET' , 'POST'])
- @app.route("/user/<int:user_id>/add_book", methods=['GET' , 'POST'])
- @app.route("/<int:book_id>/read_book")
- @app.route("/buy/<int:book_id>")
- @app.route("/<int:user_id>/lib/see_req_lib")
- @app.route("/<int:user_id>/user/<int:book_id>/approve_book")
- @app.route("/<int:user_id>/lib/sections")
- @app.route("/<int:section_id>/remove_section")
- @app.route("/lib/<int:user_id>/<int:section_id>/update_section",methods=["GET"," POST"])
- @app.route("/lib/<int:user_id>/add_section",methods=["GET","POST"])
- @app.route("/<int:user_id>/lib/books_issued")
- @app.route("/<int:user_id>/search", methods=["GET","POST"])
- @app.route("/<int:user_id>/cart")
- @app.route("/bulk_buy",methods=['GET','POST'])
- @app.route("/download",methods=['GET','POST'])
- @app.route("/<int:user_id>/create_cart")
- @app.route("/<int:user_id>/add_to_cart/<int:book_id>")
- @app.route("/<int:user_id>/remove_from_cart/<int:book_id>")
- @app.route("/<int:user_id>/lib/authors")

- @app.route("/<int:author_id>/remove_author")
- @app.route("/<int:user_id>/lib/<int:author_id>/update_author",methods=["GET","POST"])
- @app.route("/<int:user_id>/lib/add_author",methods=["GET","POST"])
- @app.route("/<int:user_id>/lib/stats")
- @app.route("/<int:user_id>/<int:book_id>/add_feedback", methods=['GET','POST'])
- @app.route("/<int:user_id>/<int:book_id>/remove_feedback")
- @app.route("/<int:user_id>/logout")
- @app.route("/about")

Database design

Models:

As per the application requirements 5 different models have been created

User: Has various attributes to store user data:

usr_id: To store user id of the user
f_name: To store first name of the user
l_name: To store last name of the user
email: To store email of the user
password: To store password of the user
role: To store role of the user
is_active: To store the login state of the user
books_owned: To get books owned the user (Many to many relationship with Book)
books_requested: To get books requested by the user (Many to many relationship with Book)
cart: To store cart details of the user (Many to many relationship with Cart)
Methods:
login: To login user into the application (set is_active to True)
logout: To logout user out of the application (set is_active to False)

Book: Has various attributes to store book data:

book_id: To store book id of the book
book_name: To store title of the book
book_description: To store description of the book
content: To store the name of the file storing the content of the book
section_id: To get section id to which the book belongs (Many to one relationship with Section)
price: To store price of the books
feedbacks: To get the feedbacks the book has (Many to many relationship with Book)

Section: Has various attributes to store the section data:

section_id: To store section id of the book
section_name: To store name of the section
section_description: To store description of the section
date_created: To store the date when the section was created
books: To get the books that the section has (One to many relationship with Book)

Author: Has various attributes to store the author data:
author_id: To store author id of the author
author_name: To store name of the author
author_description: To store description of the author
books: To get the books that have been authored by the author

Cart: Has various attributes to store the cart data:
cart_id: To store the cart id of the cart
user_id: To get the user id of the cart owner
book: To get the books in the cart

Relationship Tables:

issued_books: This is a relationship table for many to many relationship between the book and user model. This relationship is useful because one user can have many books issued and vice-versa.

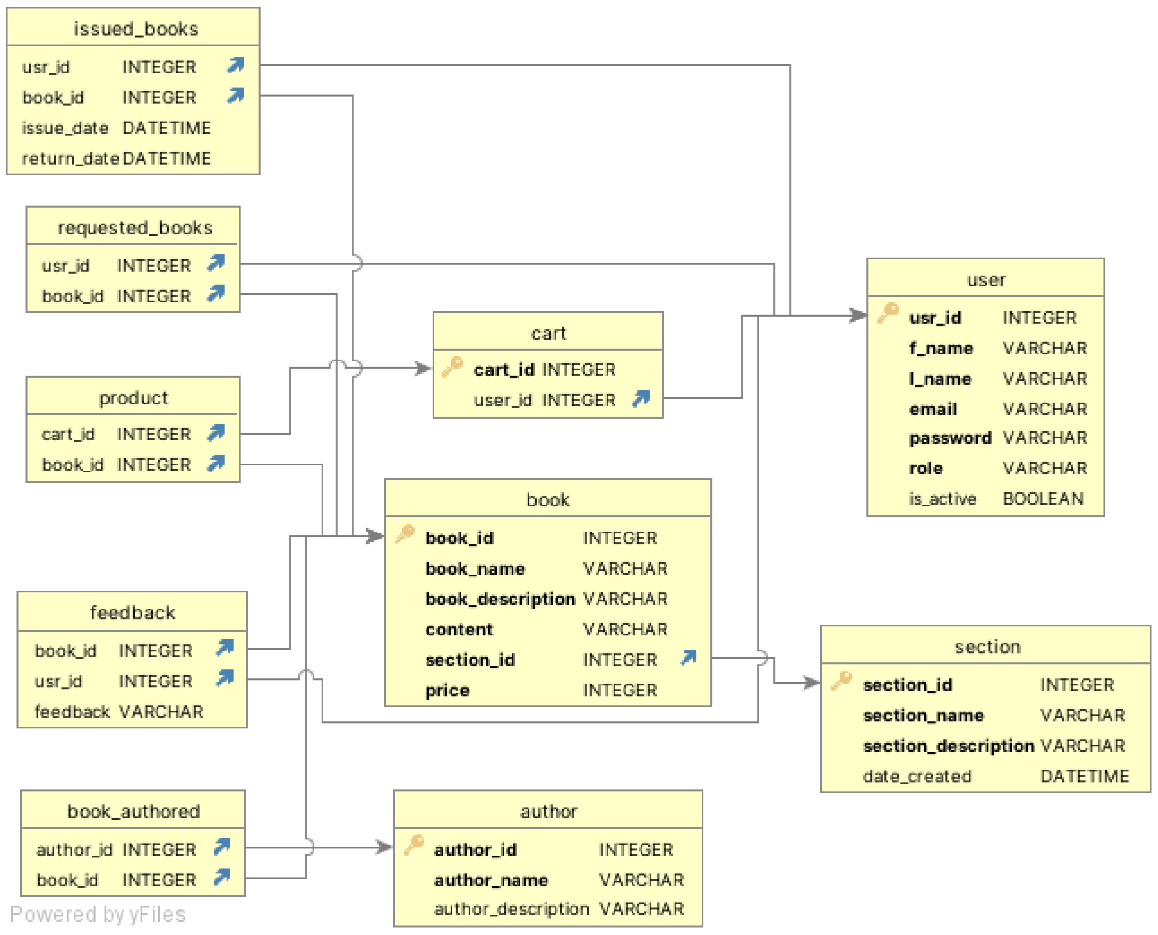
requested_books: This is a relationship table for many to many relationship between the book and user model. This relationship is useful because one user can have many books requested and vice-versa.

book_authored: This is a relationship table for many to many relationship between the book and author model. This relationship is useful because one book can have many authors and vice-versa.

product: This is a relationship table for many to many relationship between the book and cart model. This relationship is useful because one book can be in many carts and vice-versa.

feedback: This is a relationship table for many to many relationship between the book and user model. This relationship is useful because one book can give many feedbacks and vice-versa.

ER Diagram:



API Design

Four different API's have been created:

BookAPI: Implemented using get,post,put and delete methods

GET : To fetch details about all the books

PUT : To update a particular book given it's book id

POST : To add a book given it's details

DELETE : To delete a particular book given it's book id

URL's: ('/api/books', '/api/book/<int:book_id>')

AuthorAPI: Implemented using GET, PUT, POST and DELETE methods:

GET : To fetch details about all the authors
PUT : To update a particular author given it’s author id
POST : To add a author given it’s details
DELETE : To delete a particular author given it’s author id
URL’s: ('/api/authors', '/api/author/<int:author_id>')

SectionAPI: Implemented using get,post,put and delete methods
GET : To fetch details about all the sections
PUT : To update a particular section given it’s section id
POST : To add a section given it’s details
DELETE : To delete a particular section given it’s section id
URL’s: ('/api/sections', '/api/section/<int:section_id>')

GraphAPI: Implemented using get method
GET : To fetch the graphs related to statistics by specifying which graph to show
URL : ('/api/graphs/<dist>')

Project Hierarchy

```
.
└── MAD I Project - Bookit/
    ├── instance/
    │   └── project.sqlite3
    ├── static/
    │   ├── book_content_pdf/
    │   │   ├── File1.pdf
    │   │   ├── File10.pdf
    │   │   ├── File11.pdf
    │   │   ├── File12.pdf
    │   │   ├── File2.pdf
    │   │   ├── File3.pdf
    │   │   ├── File4.pdf
    │   │   ├── File5.pdf
    │   │   ├── File6.pdf
    │   │   ├── File7.pdf
    │   │   ├── File8.pdf
    │   │   └── File9.pdf
    │   ├── aboutbg.png
    │   ├── bg1.png
    │   ├── bgfinal.png
    │   ├── bgreadbook.png
    │   ├── Bookitwh.png
    │   ├── book_dist.png
    │   ├── copyright-line.png
    │   ├── home-5-line.png
    │   ├── issued_dist.png
    │   ├── libbg.png
    │   ├── logout.png
    │   ├── shopping-cart-line.png
    │   ├── sideimg.png
    │   ├── Signup.png
    │   ├── singlebook.png
    │   ├── style.css
    │   ├── user-line-black-big.png
    │   ├── user-line.png
    │   └── userbg.png
    ├── templates/
    │   ├── about_us.html
    │   ├── add_author.html
    │   ├── add_book.html
    │   ├── add_feedback.html
    │   ├── add_section.html
    │   ├── all_books.html
    │   ├── bulk_buy.html
    │   ├── buy_book.html
    │   ├── cart.html
    │   ├── home.html
    │   ├── lib_login.html
    │   ├── lib_profile.html
    │   ├── read_book.html
    │   ├── search.html
    │   ├── see_req_lib.html
    │   ├── statistics.html
    │   ├── successful.html
    │   ├── unauthorized.html
    │   ├── update_author.html
    │   ├── update_book.html
    │   ├── update_section.html
    │   ├── user_books.html
    │   ├── user_login.html
    │   ├── user_profile.html
    │   ├── user_signup.html
    │   ├── view_authors.html
    │   ├── view_book_info.html
    │   ├── view_book_issued_lib.html
    │   ├── view_req_books.html
    │   ├── view_section.html
    │   └── wrongdetails.html
```

```
├── api.py
├── app.py
└── models.py
```

As seen above the project has various different components in terms of file hierarchy. At the root level there are all the .py files that serve different purposes (controllers, models and api) then we have the static, templates and instance folder. The templates folder has all the .html files that display the main web content, the static folder has all the images, css stylesheet and a book_content_pdf folder which has all the PDF files that contain the book content and finally the instance folder has the database (project.sqlite3)

Future Prospects

In the future I am planning to work on many things which include adding a robust frontend to the application using JavaScript and JS frameworks which will enhance the UI/UX of the application, Making the security aspects more robust and stronger that will make unauthorized access to the application much harder, Adding a good PDF reader (most probably using PDF.js), Adding a text-to-speech functionality to the application, Adding some ML models to the application which can predict different trends about the user and according to them suggest books to the users etc.

Video:

MAD I App Demo.mp4